In the specification:

Please amend the paragraph beginning at page **1**, line ~~**18**~~ as follows:

An object model is a formal description of an object-oriented application. Semantic elements of an object model describe object classes, attributes of object classes, relationships between object classes and inheritance between object classes. One example object-oriented application is time-based block diagram modeling. Dynamic real-world systems such as electrical circuits, shock absorbers, braking systems, and many other electrical, mechanical and thermodynamic systems may be modeled, simulated and analyzed on a computer system using block diagram modeling. Block diagram modeling graphically depicts time-dependent mathematical relationships among a system's inputs, states and outputs, typically for display on a graphical user interface (GUI). Block diagram modeling may also be used to simulate the behavior of a system for a specified time span. Object-oriented applications include all forms of computation paradigms for block diagrams.

A1

Please amend the paragraph beginning at page **2**, line **15** as follows:

One or more of the following features may also be included. The non-critical portions are post-processing units. Post-processing units are logical units of the model that have no synchronized data outputs that feed non-post-processing sections of the model. Generating further includes establishing an inter-process communication link between the code and the non-critical portions of the model. The method may further include receiving output from the code via the inter-process communications link. The method may also include executing the code on a target processor. The method may also include processing the output in the non-critical portions of the model.

A2

Please amend the paragraph beginning at page **7**, line **10** as follows:

FIG. 1 shows ~~a~~ an exemplary system 10. The system 10 includes a host computer 12, such as a personal computer (PC). Computer 12 may be connected to a network 14, such as the Internet, that runs TCP/IP (Transmission Control Protocol/Internet Protocol) or another protocol. Connections may be via Ethernet, wireless link, or telephone line.

A3

Please amend the paragraph beginning at page **8**, line **1** as follows:

The host computer 12 communicates with a target computer 32 via a communications link 34. The target computer 32 runs a real-time operating system (RTOS) 36. The target computer 32 can also includes an input/output (I/O) port 38 for producing hardware I/O to a hardware device 40 connected to the target computer 32. The target computer 32 can be a separate computer residing within the network 14, or it can be the host computer 12 that performs the function of the target computer 32. The target computer 32 can also be a dedicated computer that is directly attached to the host computer via communication link 34, which can be shared memory or a high speed bus.

Please amend the paragraph beginning at page **8**, line **6** as follows:

The code generation process 42 executes in the host computer 12. The code generation process 42 is a process in which a behavior represented by a modeling diagram executing in computer 12 and being displayed on the GUI 28 is translated into a standalone, real-time software program code, e.g., C code. An example automatic code generator is the Target Language Compiler included in the Real-Time Workshop ~~Code generation process~~ that generates code for Simulink® block diagrams and stateflow charts from MathWorks, Inc. of Natick, Massachusetts, incorporated herein by reference. The real-time code includes only code executing in the target computer 32 that is characterized as critical for control of the hardware device 40 and not code generated for other devices (not shown) that perform off-line operations such as run-time analysis and visualization of control signals; this code would be characterized as non-critical.

Please amend the paragraph beginning at page **9**, line **5** as follows:

Every block in the block diagram model 50 is an instance of a specific type of block. The type of block determines the relationship between a block's outputs and its inputs, states, and time. The block diagram model 50 may contain any number of instances of any type of block needed to model a system. The blocks 52 are also characterized as critical real-time components of the block diagram model 50. The block diagram model 50 also includes two analysis/visualization components, i.e., a strip chart 54 and a gauge 56; these components are

characterized as non-critical. As is typical in the block diagram model 50, real-time components and analysis/visualization components are intermingled. As will be described below, code generated for the block diagram model 50 and executing on the target computer 32 only includes the critical real-time components, called the core elements, that are crucial to the control of the hardware device 40. No code is generated for devices such as the strip chart ~~54~~ 56 and gauge ~~56~~ 58 that perform off-line operations such as run-time analysis and visualization of control signals. The process 42 determines which components of the block diagram 50 are core elements and which components are non-essential elements, as is described below.

Please amend the paragraph beginning at page **10**, line **4** as follows:

Core or critical elements are computational elements of a block diagram, e.g., blocks and signals, that represent essential computations of a real world system. By essential we mean those elements that are critical to the control of the example hardware device 40. For example, controller logic is a core real-time element.

Please amend the paragraph beginning at page **11**, line **11** as follows:

The block diagram model 50 includes core elements 60 and two PPUs, i.e., PPU 62 and PPU 64. The block diagram model 50 is simulated by running all the components, core elements 60, PPU 62 and PPU 64, in an interpreted manner on the host computer 12. When a standalone, run-time program is generated for the block diagram model 50, the PPU 62 and PPU 64 are filtered or excluded from the generated software code. Specifically, no software code will execute on the target computer 32 that is non-essential or non-critical. That is, there is no software code generated for the target computer 32 to perform scaling and data smoothing operations, i.e., PPU 62, as well as the monitoring of the 'P' signal and subsequent target feedback, i.e., PPU 64. The operations performed by PPU 62 and PPU 64 are not required for the target computer 32, only for non-synchronized host/target operations such as debugging or monitoring the performance of the target computer 32. Debugging can include providing updated non-synchronized outputs to the code executing on the target computer 32. Using inter-process communication over a communication line 66, PPU 64 acquires data from the core elements 60

of the target computer 32 and run-time post processing in the PPU 64 is performed on the host computer 12. Feedback control performed by PPU 64 is realized by sending parameters or commands to the target computer 32 over the communication line 66. The communication line 66 includes a physical link such as TCP/IP, serial or shared memory, and contains messages having formats that indicate the type of message received.

Please amend the paragraph beginning at page **13**, line **1** as follows:

A slightly more refined definition of a PPU is that it is a logical unit of a block diagram model, or a logical chain of units, that perform run-time processing of target computer signals. The essential characteristic of a PPU is that it has no synchronized data outputs that feed non-PPU sections of the block diagram model 50, i.e., a PPU may only perform post processing of target computer signals.